KRIPTANALISIS MD5 DENGAN MENGGUNAKAN PENDEKATAN KOMPUTASI KINERJA TINGGI

Rizky Alfiansyah[1], Fitriyani[2], Nurul Ikhsan[3]

[123]Program Studi Ilmu Komputasi, Telkom University, Bandung
[1]rizkywingrove@gmail.com, [2]fitriyani.y@gmail.com, [3]nurul.ikhsan@yahoo.co.id

Abstrak

MD5 merupakan sebuah algoritma kriptografi *hash function* yang banyak digunakan sebagai *digital signature* dari sebuah *file* atau sebagai enkripsi *password* dalam *database*. Salah satu teknik kriptanalisis yang bisa diterapkan untuk menembus enkripsi MD5 adalah *exhaustive key search*. Kebutuhan performa komputasi tingkat tinggi dari teknik ini akan diatasi dengan penggunaan dua buah GPU kelas *high-end* (NVIDIA & AMD), dengan kriptanalisis yang diimplementasikan secara paralel dengan menggunakan bahasa CUDA dan OpenCL.

Pengujian dilakukan dengan menggunakan 1 s/d 9 digit *random string* yang berdasar dari 65 macam karakter. Hasil pengujian menunjukkan sebuah *high-end* GPU memiliki batas kemampuan kriptanalisis hingga 8 s/d 9 digit *random string*, dengan waktu kriptanalisis terlama mencapai lebih dari 1 minggu. Sedangkan untuk perbandingan performansi, OpenCL pada GPU AMD menghasilkan performa terbaik jika dibandingkan dengan CUDA & OpenCL pada GPU NVIDIA.

Kata kunci : MD5, Kriptanalisis, CUDA, OpenCL, GPU.

Abstract

MD5 is a hash function that mostly used as a digital signature from a file or as an encryption when passwords being stored in a database. One way to decrypt a MD5 hash is by doing an exhaustive key search attack. This kind of attack is relatively easy to implement, but requires some hardwares with high computational performance. Two high-end GPUs (NVIDIA & AMD) is proposed to run a parallel approach of this attack, with the use of two programming languages, CUDA & OpenCL.

1 to 9 digits of random string that based on 65 kind of characters, is being used to do experiments in this research. Experiment results showed that single high-end GPU can only handle the cryptanalysis for at most 9 digits of random string, with the longest running time reached 1 week or more. In terms of performance comparisons, OpenCL on AMD GPU gave the best performance when being compared to CUDA and OpenCL on NVIDIA GPU.

Keywords : MD5, Cryptanalysis, CUDA, OpenCL, GPU.

1. Pendahuluan

Keamanan data dan informasi sudah menjadi bagian penting pada zaman internet ini. Banyaknya data dan informasi pada jaringan internet atau pada sebuah jaringan komputer rentan dimanfaatkan oleh pihak yang tidak bertanggung-jawab. Salah satu tindakan untuk mencegah hal ini adalah dengan menggunakan enkripsi pada data, atau bisa juga disebut kriptografi. Untuk mengetahui keamanan enkripsi, perlu dilakukan serangan pada eknripsi tersebut, atau bisa juga disebut kriptanalisis.

Pada penelitian ini penulis mencoba untuk mengimplementasikan sebuah teknik *brute-force* untuk memecahkan sebuah algoritma kriptografi yaitu *Message Digest Algorithm 5* (MD5). Teknik yang akan diimplementasikan adalah *exhaustive key search*. Teknik ini tergolong mudah untuk diimplementasikan, tetapi memiliki konsekuensi yaitu kebutuhan komputasi yang besar. Dalam mengatasi konsekuensi ini, penulis akan menggunakan pendekatan yang berbeda.

Dengan munculnya banyak ilmu atau teknik komputasi paralel serta teknologi perangkat keras yang semakin mendukung, maka kriptanalisis bukan lagi hal yang sulit dilakukan secara komputasi. Dahulu kriptanalisis lekat dengan *effort* komputasi yang sangat besar dan membutuhkan waktu *cracking* sampai berhari-hari untuk menembus 4 s/d 8 digit *password*[1][6]. Tetapi karena sekarang *high performance computing* sudah semakin terjangkau, dalam bentuk GPU, maka penulis akan menjadikan hal tersebut sebagai pendekatan untuk mendukung implementasi kriptanalisis pada penelitian ini.

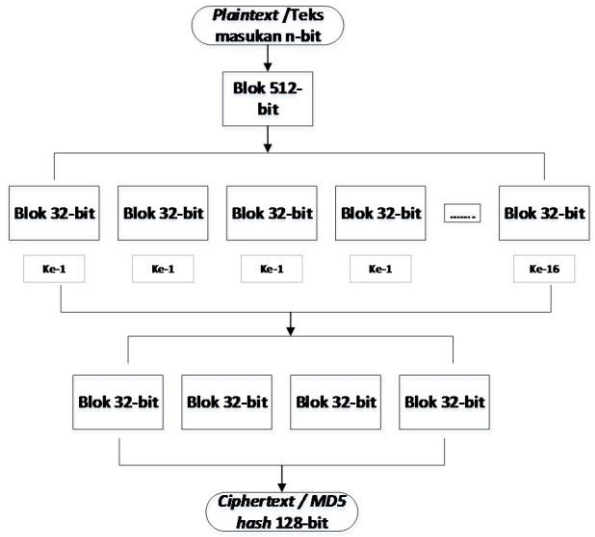
2. Tinjauan Pustaka

2.1 Algoritma Kriptografi

Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut. Algoritma kriptografi umumnya terdiri dari 3 fungsi dasar, yaitu enkripsi, kunci, dan dekripsi. Berdasarkan pemakaian kunci, ada 3 macam algoritma kriptografi, yaitu algoritma simetri, algoritma asimetri, dan *hash function*. Algoritma *hash function* tidak memiliki kunci sehingga tidak ada cara untuk mengembalikan *ciphertext* ke bentuk *plaintext*.

2.2 *Message Digest Algorithm* 5

MD5 termasuk dalam kategori algoritma *hash function*, yang berarti hasil enkripsi MD5 tidak bisa dikembalikan ke bentuk semula. informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan khusus. MD5 sering digunakan sebagai *digital signature* dari sebuah *file* , atau sebagai bentuk pengamanan dari *password* yang tersimpan dalam *database*. MD5 mengolah teks masukan n bit yang dipadatkan ke bentuk blok 512 bit. Lalu blok ini dibagi dalam 16 bentuk sub-blok yang masing-masing berukuran 32 bit dan diproses menggunakan operasi-operasi pada algoritma MD5. Keluaran terdiri dari 4 blok berukuran 32 bit, yang digabungkan menjadi panjang 128 bit. Gambaran proses ini bisa dilihat pada Gambar 2-1.



Gambar 2-1: Proses MD5.

Contoh keluaran enkripsi MD5:
"coba" = c3ec0f7b054e729c5a716c8125839829
"cobain" = bed5022921aa27b94229bb951668c58b

2.3 Kriptanalisis

Atau biasa juga disebut serangan, merupakan sebuah kegiatan untuk menembus sebuah enkripsi, yang bisa dilakukan dengan berbagai cara. Teknik serangan yang sering digunakan dalam kegiatan kriptanalisis terbagi dalam bentuk, yaitu *ciphertext-only attack*, *known-plaintext attack*, dan *defined-plaintext attack*.

2.4 *Exhaustive Key Search*

Teknik ini merupakan variasi dari *brute-force attack* dan masuk pada kategori *ciphertext-only attack*. *Exhaustive key search* dipilih karena ketika melakukan kriptanalisis MD5, kita dihadapkan pada kondisi hanya mengetahui *ciphertext*. Ruang pencarian *exhaustive key search* pada sebuah *hash* MD5 bisa mencapai 2^{128} kemungkinan.

2.5 GPU

GPU yang digunakan pada penelitian ini adalah AMD R9 270x dan NVIDIA GTX 770. Secara arsitektur, kedua GPU ini termasuk berimbang dan tentunya lebih bagus performanya jika dibandingkan dengan CPU, seperti terlihat pada Gambar 2-2.

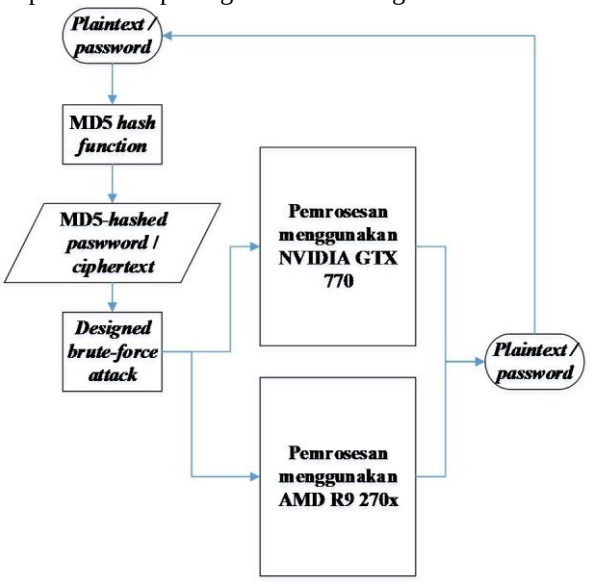
Gambar 2-2: Arsitektur CPU & GPU

CUDA dan OpenCL digunakan untuk mengimplementasikan kriptanalisis secara paralel pada masing-masing GPU. Sehingga akan menghasilkan 3 pendekatan: CUDA & OpenCL pada GPU NVIDIA serta OpenCL pada GPU AMD.

3. Perancangan Sistem

3.1 *Flowchart* Sistem yang akan Dibangun

Rancangan sistem yang dibangun untuk melakukan kriptanalisis dapat digambarkan sebagai berikut.



Gambar 3-1 Rancangan Sistem

Berdasarkan Gambar 3-1, dapat dilihat sistem yang dirancang pada penelitian ini. Secara garis besar sistem ini terdiri dari 3 proses utama.

a. *Input*
   Pada tahap ini dilakukan proses input hasil enkripsi dari *plaintext* yang telah diproses dengan menggunakan metode MD5 sehinggan menghasilkan sebuah MD5 *hash*.
b. Kriptanalisis
   Pada tahap ini *ciphertext* di-kriptanalisis dan dijalankan pada sub-proses, yaitu 3 pendekatan

yang sudah disebutkan sebelumnya, untuk mendapatkan *plaintext*. Dengan menggunakan strategi serangan yang dirancang, semua kemungkinan *hash* akan di-*generate* dan dibandingkan dengan *hash* pada *input*.

c. *Output*
Jika pada proses sebelumnya ada *hash* hasil *generate* yang sama dengan *hash input*, maka *plaintext* berhasil ditemukan.

3.2 Implementasi

3.2.1 Strategi Penyerangan

Karena MD5 merupakan enkripsi yang bersifat *one-way hash function*, maka implementasi *brute-force* yang paling efektif adalah dengan membandingkan. *Generate random string* (berdasarkan set karakter yang sudah didefinisi-kan) dan enkripsi tiap hasil *generate* tersebut sehingga menghasilkan MD5 hash dari tiap-tiap hasil *generate* tersebut. Lalu bandingkan dengan *hash* yang menjadi *input* dari program (hash yang akan di-*crack*). Untuk performansi, ada beberapa parameter pada program yang sudah didefinisikan semenjak awal program dibuat, dan tidak akan berubah walaupun pendekatan yang digunakan berbeda, yaitu:

- *Threads/work-items* sebanyak 128
- *Blocks/work-groups* sebanyak 64
- *MD5_per_kernel* sebesar 200

3.2.2 Algoritma Serangan (Paralel)

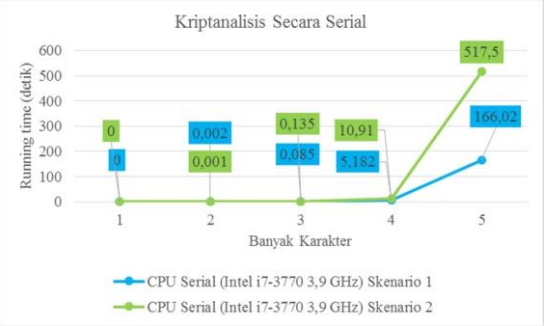
Pada ketiga program yang dibuat, ada sebuah *kernel* yang dieksekusi secara paralel dan tidak akan berubah selama penelitian dilakukan. *Kernel* ini merupakan terjemahan dari strategi penyerangan yang sudah dijelaskan sebelumnya. Dalam *pseudocode*, *kernel* tersebut adalah:

```
for i=0 to md5_per_kernel
        for j=0  to digit
                word[i]-> bruteCharSet[i]

        end for
        md5_verify(word)
        if word = userinput
                for k=0 to max_digit

        correctpass[k]=word[k]
                correctpass[totalLen]=0
                end for
        else
                bruteIncrement()
        end if
end for
```

4. Hasil Pengujian dan Analisis

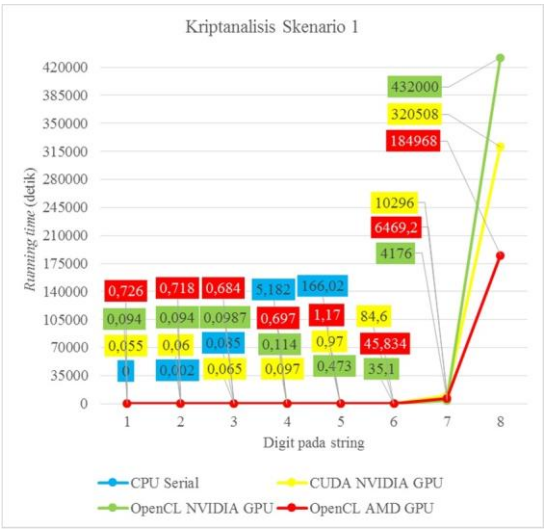
Pengujian kriptanalisis MD5 dilakukan dengan menggunakan *random string* 1 s/d 4 karakter untuk memperlihatkan pola kriptanalisis, dan studi kasus *password string* yang bervariasi dari 4 s/d 9 karakter. Setiap kriptanalisis *password hash* (*input* program) akan diuji sebanyak 3 kali dan waktu *running* program yang dicatat merupakan rata-rata dari ketiga pengujian tersebut.

Ada dua skenario yang diuji pada tahap pengujian ini. Skenario pertama adalah kondisi dimana *input string* hanya mengandung kombinasi *lowercase*, *uppercase*, dan *numerical*. Sedangkan skenario kedua adalah kondisi dimana *input string* merupakan kombinasi *lowercase*, *uppercase*, numerical, dan *symbols*. Berdasarkan kedua skenario ini, *input* akan di-kriptanalisis dengan 3 pendekatan yang berbeda: CUDA NVIDIA, OpenCL NVIDIA, dan OpenCL AMD.



Gambar 4-1: Hasil pengujian secara serial.

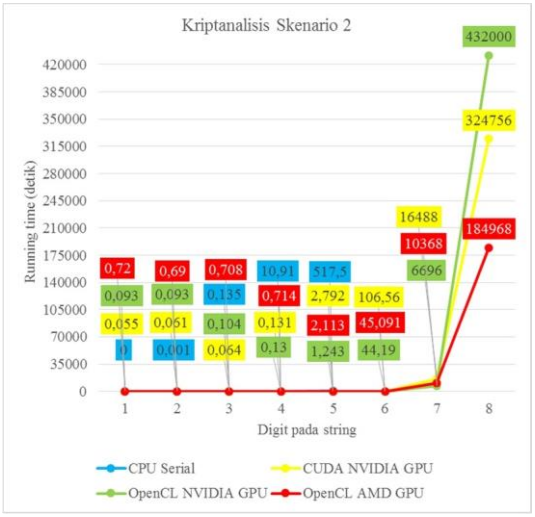
Pada Gambar 4-1, mulai terlihat peningkatan waktu *running* yang cukup signifikan pada kriptanalisis 4 dan 5 karakter. Ada 2 hal yang menyebabkan ini terjadi. Yang pertama adalah karena perbedaan skenario yang menyebabkan ruang pencarian *password* meluas. Sedangkan yang kedua adalah karena cara pendefinisian set karakter dalam program, berpengaruh pada lama *running* program.



Gambar 4-2: Kriptanalisis skenario 1.

Sebelumnya disebutkan bahwa banyak karakter *password* yang akan diuji pada penelitian ini adalah 4 s/d 9 karakter. Berdasarkan tren dan pola waktu hasil *running* program selama pengujian, contoh pada Gambar 4-2, terdapat 3 rentang waktu *cracking* pada penelitian ini:

- Yang bisa di-*crack* dengan waktu < 5 jam : *password* dengan 7 karakter atau kurang.
- Yang bisa di-*crack* dengan waktu < 4 hari: *password* dengan 8 karakter.
- Yang bisa di-*crack* tetapi harus mengeluarkan biaya komputasi yang lebih mahal dari sekedar sebuah GPU dan dengan waktu yang lebih dari 1 minggu: *password* dengan 9 karakter atau lebih.



Gambar 4-3: Kriptanalisis skenario 2.

Berdasarkan pengujian di penelitian ini, contoh pada Gambar 4-3, untuk masalah *password cracking*, *hashing*, atau *brute-forcing* secara umum, perangkat AMD R9 270x lebih layak untuk digunakan. Ada beberapa faktor yang membuat hal ini terjadi karena segi jumlah *thread processors*, kedua GPU tidak jauh berbeda (GTX 770 sedikit lebih banyak), tetapi R9 270x memiliki *compute unit* 2,5x lebih banyak dari GTX 770. Untuk *password cracking* atau *hashing* yang murni mengandalkan banyaknya prosesor (ALU) yang berjalan, tentu R9 270x memiliki performa yang lebih tinggi jika dibandingkan dengan NVIDIA GTX 770.

Pada kondisi tertentu, OpenCL pada kedua perangkat juga lebih layak digunakan untuk mengimplementasikan kriptanalisis jika dibandingkan dengan CUDA. Mengingat CUDA merupakan bahasa eksklusif dari NVIDIA, dapat diprediksi jika performa CUDA seharusnya lebih bagus dari OpenCL ketika kedua bahasa ini diimplementasikan pada perangkat NVIDIA. Tetapi bila dilihat pada Gambar 4-3, OpenCL pada NVIDIA memiliki performansi kriptanalisis terbaik hingga 7 karakter. Anomali ini dapat dijelaskan setelah mengetahui kemampuan *compiler* pada masing-masing bahasa CUDA dan OpenCL[9].

Program kriptanalisis yang dibuat sangat mengandalkan 2 aspek pada kedua perangkat yang digunakan, banyaknya ALU yang bisa berjalan bersamaan, dan juga ukuran serta kecepatan *memory buffer* yang bisa digunakan ketika melakukan proses *hashing* dan *comparing*. Dengan mengetahui kedua aspek ini dan melihat perbandingan pada bagian *Data Movement*[9], maka kondisi seperti yang sudah diutarakan sebelumnya, dapat dijelaskan.

Ketika *kernel* menjalankan fungsi *cracking*, ratusan hingga jutaan *hash* dihasilkan dan dibandingkan, tentunya proses ini akan menggunakan *local*, *shared*, dan *global memory*. Pada *compiler* CUDA, *data movement* ketika melakukan *cracking* membutuhkan 3x instruksi lebih banyak daripada *compiler* OpenCL.

Tetapi ketika memasuki kriptanalisis 8 karakter, performansi OpenCL NVIDIA melambat dan waktu kriptanalisis meningkat sampai lebih dari 4 hari, dan kehabisan memori di hari ke-5. Hal ini bisa terjadi karena ketika komputasi membutuhkan *resource* lebih dari yang disediakan, manajemen memori OpenCL pada NVIDIA tidak sama optimal ketika menggunakan CUDA pada NVIDIA.

5. Penutup

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian tugas akhir ini adalah sebagai berikut.

1) Algoritma *brute-force* yang menjadi dasar dari kriptanalisis MD5 berhasil diterapkan secara paralel dengan menggunakan 2 bahasa, yaitu CUDA dan OpenCL. Kedua bahasa ini dijalankan pada NVIDIA GPU & AMD GPU, sehingga menghasilkan 3 pendekatan: CUDA & OpenCL pada NVIDIA serta OpenCL pada AMD.
2) Untuk masalah komputasi yang murni mengandalkan manajemen memori pada *hardware* dan juga banyaknya ALU atau operasi yang bisa berjalan bersamaan secara paralel, maka pendekatan OpenCL yang dijalankan pada perangkat AMD GPU adalah pilihan terbaik.
3) Batas kemampuan sebuah *high-end* GPU untuk melakukan kriptanalisis hanya sampai ke 8 digit karakter dengan waktu running 2 s/d 4 hari.

5.2 Saran

Saran yang bisa diperhatikan jika ingin mengembangkan penelitian tugas akhir ini adalah sebagai berikut.

1) Mencoba berbagai kemungkinan pengembangan kriptanalisis, seperti menambahkan set karakter dan digit yang akan di-*crack*, dan memakai perangkat yang lebih tinggi performanya.
2) Memperbaiki *performance tuning* dalam program, seperti memodelkan banyak *threads* dan *blocks* yang paling optimal untuk melakukan implementasi kriptanalisis ini.

3) Mengembangkan program kriptanalisis di penelitian ini untuk menembus enkripsi *salted-password*.

Daftar Pustaka

[1] Nguyen, Duc H., Nguyen, Thuy T., Duong, Tan N., Pham, Phong H., 2008, *Cryptanalysis of MD5 on GPU Cluster*, Vietnam, Hanoi University of Science and Technology.

[2] Ariyus, Doni., 2008, *Pengantar Ilmu Kriptografi*, D.I. Yogyakarta, STMIK AMIKOM Yogyakarta.

[3] Narayanan, Arvind., Shmatikov, Vitaly., 2005, *Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff*, USA, University of Texas.

[4] Weir, Matt., Aggarwal, Sudhir., de Medeiros, Breno., Glodek, Bill., 2009, *Password Cracking Using Probabilistic Context-Free Grammars*, USA, Florida State University.

[5] Schaefer, Edward., 2014, *An Introduction to Cryptography and Cryptanalysis*, USA, Santa Clara University.

[6] Sprengers, Martijn., 2010, *GPU-based Password Cracking*, Netherlands, Radboud University Nijmegen.

[7] Kaushik, Kumar., 2008, *Cluster Computing*, India, Cochin University Of Science and Technology.

[8] Tittel, Ed., 2005, *HPC For Dummies*, Canada, John Wiley & Sons Canada, Ltd.

[9] Fang, Jianbin., Varbanescu, Ana L., Sips, Henk., *A Comprehensive Performance Comparison Of CUDA and OpenCL*, Netherlands, Delft University of Technology.

[10] Springer, Paul., *Berkeley Dwarfs On Cuda*., Germany, RWTH Aachen University.